Removing Channel Estimation by Location-Only Based Deep Learning for RIS Aided Mobile Edge Computing

Xiaoyan Hu, Christos Masouros, and Kai-Kit Wong
Department of Electronic and Electrical Engineering, University College London, London, UK
Email:{xiaoyan.hu, c.masouros, kai-kit.wong}@ucl.ac.uk

***Abstract*—In this paper, we investigate a deep learning architecture for lightweight online implementation of a reconfigurable intelligent surface (RIS)-aided multi-user mobile edge computing (MEC) system, where the optimized performance can be achieved based on user equipment's (UEs') location-only information. Assuming that each UE is endowed with a limited energy budget, we aim at maximizing the total completed task-input bits (TCTB) of all UEs within a given time slot, through jointly optimizing the RIS reflecting coefficients, the receive beamforming vectors, and UEs' energy partition strategies for local computing and computation offloading. Due to the coupled optimization variables, a three-step block coordinate descending (BCD) algorithm is first proposed to effectively solve the formulated TCTB maximization problem iteratively with guaranteed convergence. The location-only deep learning architecture is then constructed to emulate the proposed BCD optimization algorithm, through which the pilot channel estimation and feedback can be removed for online implementation with low complexity. The simulation results reveal a close match between the performance of the BCD optimization algorithm and the location-only data-driven architecture, all with superior performance to existing benchmarks.**

***Index Terms*—Mobile edge computing, reconfigurable intelligent surface, location-only, deep learning.**

I. Introduction

Massive connectivity is one of the major performance goals of 5G and beyond technologies, fuelled by the proliferation of IoT devices. Driven by extensive emerging computation-intensive applications, the computing demands for massive user equipment (UEs) are witnessing an unprecedented growth. In order to liberate the resource-limited UEs from heavy computation workloads and provide them with high-performance low-latency computing services, mobile edge computing (MEC) promotes the use of cloud computing capabilities at the edge of mobile networks through integrating MEC servers at the wireless access points (APs) [1].

The performance enhancement of MEC in various wireless networks has been confirmed in existing works [2–6]. In order to further enhance the uplink offloading performance, the technology of unmanned aerial vehicle (UAV) communications was introduced into the MEC systems [7–9]. However, the traditional battery-based UAVs are also energy-limited with considerable propulsion energy cost. Recently, great attentions have been drawn to the technology of reconfigurable intelligent surface (RIS), due to its advantages of low cost, easy deployment, and fine-grained passive beamforming [10–12]. Several pioneer RIS-aided MEC works have explored the benefits of utilizing RISs in MEC systems [13–15], and it was verified that significant performance improvement can be attained compared with the case without RIS.

For RIS-aided MEC works, iterative algorithms are usually necessary for jointly optimizing the resource allocation and the RIS coefficients. Although iterative algorithms may be capable of providing near-optimal solutions, the high computational complexity and signalling overheads may hider their use in practical networks. To tackle this issue, deep learning architectures provide a promising way to achieve offline training and online implementation [16]. Note that deep learning methods have been investigated in several MEC systems to simplify the optimization algorithm [17–19]. Recently, deep reinforcement learning was also used in several RIS-aided systems [20, 21].

However, the potential of deep learning methods in simplifying the optimization algorithms of complex RIS-aided MEC systems have not been explored in the existing literature. In this paper, a multi-user RIS-aided MEC architecture is considered, where the RIS is installed to constructively control the interference and enhance the overall performance of UEs. We first propose an iterative optimization algorithm to efficiently solve the formulated problem with guaranteed convergence. As a step further, we construct a deep learning architecture employs, not the channel state information, but the location information of the UEs, readily available through GPS, to facilitate online implementations of the proposed algorithm with low complexity. The complicated pilot channel estimation and feedback can be removed when utilizing this location-only data-driven approach for online implementation.

II. System Model and Problem Formulation

We consider a RIS-aided MEC system, which consists of one M-antenna AP, one RIS with K reflecting elements, and N single-antenna ground UEs. Through choosing a desirable location of the RIS, line-of-sight (LoS) connections between the RIS and the AP, UEs can be achieved within a certain serving area. Each UE $n \in \mathcal{N} = \{1, 2, \cdots, N\}$ has extensive computation task-input bits, but with a limited energy budget dedicated for completing these task bits, denoted as E_n in Joules (J). C_n is number of CPU cycles required for completing 1-bit of UE n's input data. We adopt the partial offloading mode to handle UEs' computation tasks with parallel local computing and computation offloading [1]. The grid-powered AP is co-located with a powerful MEC server for helping UEs compute their offloaded tasks and download UEs' computation results, both in negligible time. Our aim is to maximize the total completed task-input bits (TCTB) of UEs during a given time slot T, which is equivalent to maximize the energy efficiency of UEs considering the fixed energy budgets [22].

We first introduce a partition parameter $a_n \in [0, 1]$ for UE $n \in \mathcal{N}$, then $a_n E_n$ and $(1 - a_n)E_n$ J of energy will be used for computation offloading and local computing, respectively. In this case, the offloading power of UE n is given as $p_n = \frac{a_n E_n}{T} \triangleq a_n \widetilde{E}_n$ with $\widetilde{E}_n = E_n/T$. Let s_n denote the task-input data-bearing signal transmitted by UE n for computation offloading with $|s_n| = 1$. Note that all the UEs with offloading

requirements transmit their signals simultaneously within the given time slot, and thus we can express the corresponding received signal $\mathbf{y} \in \mathbb{C}^{M\times 1}$ at the AP as

$$\mathbf{y} = \sum_{n=1}^{N} \mathbf{H}_{\mathrm{AP}} \mathbf{\Phi} \mathbf{h}_{\mathrm{r},n} \sqrt{p_n} s_n + \mathbf{n}, \tag{1}$$

where $\mathbf{h}_{\mathrm{r},n} \in \mathbb{C}^{K\times 1}$ is the relay channel between UE n and the RIS and $\mathbf{H}_{\mathrm{AP}} \in \mathbb{C}^{M\times K}$ represents the channel between RIS and the AP.[1] $\mathbf{\Phi} = \mathrm{diag}\{\boldsymbol{\phi}\}$ indicates the reflection-coefficient matrix of the RIS, where $\boldsymbol{\phi} = [\phi_1, \cdots, \phi_K]^{\mathrm{T}}$ and $\phi_k = e^{j\theta_k}$ being the phase shift of the k-th reflecting element with $\theta_k \in [0, 2\pi]$ for $k \in \mathcal{K} = \{1, 2, \cdots, K\}$. Also, $\mathbf{n} \sim \mathcal{CN}(0, \sigma^2 \mathbf{I}_M)$ is the the additive white Gaussian noise (AWGN) at the AP. The linear receive beamforming vector $\mathbf{w}_n \in \mathbb{C}^{M\times 1}$ is then used at the AP to decode the transmit signal of UE n, thus the estimated signal for UE n can be given as

$$\widehat{s}_n = \mathbf{w}_n^{\mathrm{H}} \mathbf{y} = \mathbf{w}_n^{\mathrm{H}} \sum_{n=1}^{N} \mathbf{H}_{\mathrm{AP}} \mathbf{\Phi} \mathbf{h}_{\mathrm{r},n} \sqrt{p_n} s_n + \mathbf{w}_n^{\mathrm{H}} \mathbf{n}, \forall n \in \mathcal{N}. \tag{2}$$

Then we can obtain the uplink signal-to-interference-plus-noise ratio (SINR) for offloading UE n's tasks as follows

$$\gamma_n(\mathbf{a}, \mathbf{w}_n, \boldsymbol{\phi}) = \frac{a_n \widetilde{E}_n |\mathbf{w}_n^{\mathrm{H}} \mathbf{H}_{\mathrm{AP}} \mathbf{\Phi} \mathbf{h}_{\mathrm{r},n}|^2}{\sum\limits_{i=1, i\neq n}^{N} a_i \widetilde{E}_i |\mathbf{w}_n^{\mathrm{H}} \mathbf{H}_{\mathrm{AP}} \mathbf{\Phi} \mathbf{h}_{\mathrm{r},i}|^2 + \sigma^2 ||\mathbf{w}_n^{\mathrm{H}}||^2}, \ \forall n \in \mathcal{N}, \tag{3}$$

where we denote an energy partition vector $\mathbf{a} = [a_1, \cdots, a_N]$. Then, the CTB of UE n through computation offloading is

$$R_n^{\mathrm{off}}(\mathbf{a}, \mathbf{w}_n, \boldsymbol{\phi}) = BT \log_2(1 + \gamma_n(\mathbf{a}, \mathbf{w}_n, \boldsymbol{\phi})), \ \forall n \in \mathcal{N}, \tag{4}$$

where B is the given bandwidth shared with all UEs.

As for the case of local computing, the dynamic voltage and frequency scaling (DVFS) technique is adopted at all UEs for increasing the computation energy efficiency through adaptively controlling the CPU frequency used for computing. Thus, the energy consumption of UE $n \in \mathcal{N}$ can be expressed as $T\kappa_n f_n^3$, where κ_n is the effective capacitance coefficient of UE n, and f_n is the CPU frequency of its processing server. Also, we have $(1 - a_n)E_n = T\kappa_n f_n^3$, then we can calculate $f_n = \sqrt[3]{\frac{(1-a_n)E_n}{T\kappa_n}}$. Hence, the CTB of UE n for local computing can be given as

$$R_n^{\mathrm{loc}}(a_n) = \frac{f_n T}{C_n} = \frac{T}{C_n} \sqrt[3]{\frac{(1 - a_n)\widetilde{E}_n}{\kappa_n}}, \ \forall n \in \mathcal{N}. \tag{5}$$

The corresponding TCTB maximization problem is formulated as problem (P0) given below

$$(\mathrm{P0}): \max_{\mathbf{a}, \mathbf{W}, \boldsymbol{\phi}} \ \sum_{n=1}^{N} \left(R_n^{\mathrm{off}}(\mathbf{a}, \mathbf{w}_n, \boldsymbol{\phi}) + R_n^{\mathrm{loc}}(a_n) \right) \tag{6a}$$

$$\text{s.t.} \quad a_n \in [0, 1], \ \forall n \in \mathcal{N}, \tag{6b}$$

$$|\phi_n| = 1, \ \forall n \in \mathcal{N}, \tag{6c}$$

where the objective TCTB is maximized through jointly optimizing the reflection coefficients in $\boldsymbol{\phi}$, the receive beamforming vectors in $\mathbf{W} = [\mathbf{w}_1, \cdots, \mathbf{w}_N]$, and the energy partition parameters in $\mathbf{a}$. This is a non-convex optimization problem since the parameters $\boldsymbol{\phi}$, $\mathbf{W}$, and $\mathbf{a}$ are strongly coupled. Next, we will leverage the BCD optimizatiton algorithm to solve the original problem (P0) in three steps by solving three sub-problems iteratively, as shown in the following section.

[1] It is assumed that the direct links between the UEs and the AP are negligible due to severely degradation caused by blockage.

III. Optimization Design based on CSI

A. RIS Reflecting Coefficients Design

In the χ-th ($\chi = 1, 2, \cdots$) iteration of the BCD algorithm, we first design the RIS reflecting coefficients, i,e, $\boldsymbol{\phi}$, with given $\mathbf{W} = \mathbf{W}_{\chi-1}$ and $\mathbf{a} = \mathbf{a}_{\chi-1}$. Then the RIS's reflecting coefficients design problem (P1) can be described as

$$(\mathrm{P1}): \max_{\boldsymbol{\phi}} \ \sum_{n=1}^{N} \log_2(1 + \gamma_n(\boldsymbol{\phi})) \tag{7a}$$

$$\text{s.t.} \quad |\phi_k| = 1, \ \forall k \in \mathcal{K}, \tag{7b}$$

which is still non-convex and difficult to deal with directly. According to the expression of $\gamma_n(\boldsymbol{\phi})$ in (3) for $n \in \mathcal{N}$, we can re-express $|\mathbf{w}_n^{\mathrm{H}} \mathbf{H}_{\mathrm{AP}} \mathbf{\Phi} \mathbf{h}_{\mathrm{r},i}|^2$ as

$$\begin{aligned} &|\mathbf{w}_n^{\mathrm{H}} \mathbf{H}_{\mathrm{AP}} \mathbf{\Phi} \mathbf{h}_{\mathrm{r},i}|^2 = |\mathbf{w}_n^{\mathrm{H}} \mathbf{H}_{\mathrm{AP}} \mathrm{diag}(\mathbf{h}_{\mathrm{r},i}) \boldsymbol{\phi}|^2 \\ =&|\mathbf{h}_{\mathrm{r},n,i}^{\mathrm{RIS}} \boldsymbol{\phi}|^2 = \boldsymbol{\phi}^{\mathrm{H}} (\mathbf{h}_{\mathrm{r},n,i}^{\mathrm{RIS}})^{\mathrm{H}} \mathbf{h}_{\mathrm{r},n,i}^{\mathrm{RIS}} \boldsymbol{\phi}, \\ =&\mathrm{Tr}\left(\mathbf{Q}_{n,i} \mathbf{\Psi}\right), \ \forall n, i \in \mathcal{N}, \end{aligned} \tag{8}$$

where $\mathbf{Q}_{n,i} = (\mathbf{h}_{\mathrm{r},n,i}^{\mathrm{RIS}})^{\mathrm{H}} \mathbf{h}_{\mathrm{r},n,i}^{\mathrm{RIS}}$ with $\mathbf{h}_{\mathrm{r},n,i}^{\mathrm{RIS}} = \mathbf{w}_n^{\mathrm{H}} \mathbf{H}_{\mathrm{AP}} \mathrm{diag}(\mathbf{h}_{\mathrm{r},i}) \in \mathbb{C}^{1\times K}$ and $\mathbf{\Psi} = \boldsymbol{\phi}\boldsymbol{\phi}^{\mathrm{H}} \in \mathbb{C}^{K\times K}$ is a positive semidefinite matrix (PSD) related to the RIS coefficients.

Note that each added item in the objective function, i.e., $\log_2(1 + \gamma_n(\boldsymbol{\phi}))$, can be re-written as

$$\begin{aligned} \log_2(1 + \gamma_n(\boldsymbol{\phi})) =& \log_2 \left(\sum_{j=1}^{N} p_j \mathrm{Tr}\left(\mathbf{Q}_{n,i} \mathbf{\Psi}\right) + \sigma^2 ||\mathbf{w}_n^{\mathrm{H}}||^2 \right) \\ &- \log_2 \left(\sum_{i=1, i\neq n}^{N} p_i \mathrm{Tr}\left(\mathbf{Q}_{n,i} \mathbf{\Psi}\right) + \sigma^2 ||\mathbf{w}_n^{\mathrm{H}}||^2 \right) \\ \triangleq& F_{1,n}(\mathbf{\Psi}) - F_{2,n}(\mathbf{\Psi}), \ \forall n \in \mathcal{N}, \end{aligned} \tag{9}$$

where $F_{1,n}(\mathbf{\Psi})$ and $F_{2,n}(\mathbf{\Psi})$ are two concave functions w.r.t. $\mathbf{\Psi}$. Hence, problem (P1) can be equivalently transformed into

$$(\widetilde{\mathrm{P1}}): \max_{\mathbf{\Psi} \succeq \mathbf{0}} \ \sum_{n=1}^{N} F_{1,n}(\mathbf{\Psi}) - F_{2,n}(\mathbf{\Psi}) \tag{10a}$$

$$\text{s.t.} \quad \mathbf{\Psi}_{k,k} = 1, \ \forall k = 1, 2, \cdots, K, \tag{10b}$$

$$\mathrm{rank}(\mathbf{\Psi}) = 1. \tag{10c}$$

Next, we will leverage the DC programming [23] to effectively address the issues of the non-convex objective function in (10a) and the rank-one constraint (10b).

As for the objective function, in the $(l+1)$-th ($l = 0, 1, \cdots$) iteration of the DC programming, the second concave item, i.e., $F_{2,n}(\mathbf{\Psi})$ for $n \in \mathcal{N}$, can be approximated by its liner upper bound at the point $\mathbf{\Psi}^{(l)}$ (the solution obtained from the previous l-th iteration), which is given as

$$F_{2,n}(\mathbf{\Psi}) \leq \widehat{F}_{2,n}(\mathbf{\Psi}; \mathbf{\Psi}^{(l)}) = F_{2,n}(\mathbf{\Psi}^{(l)}) + \frac{\sum\limits_{i=1, i\neq n}^{N} p_i \left\langle (\mathbf{\Psi} - \mathbf{\Psi}^{(l)}), \nabla_{\mathbf{\Psi}} \mathrm{Tr}(\mathbf{Q}_{n,i} \mathbf{\Psi})|_{\mathbf{\Psi} = \mathbf{\Psi}^{(l)}} \right\rangle}{\ln 2 \left(\sum\limits_{i=1, i\neq n}^{N} p_i \mathrm{Tr}(\mathbf{Q}_{n,i} \mathbf{\Psi}^{(l)}) + \sigma^2 ||\mathbf{w}_n^{\mathrm{H}}||^2 \right)}, \tag{11}$$

where $\nabla_{\mathbf{\Psi}}\mathrm{Tr}(\mathbf{Q}_{n,i}\mathbf{\Psi})|_{\mathbf{\Psi}=\mathbf{\Psi}^{(l)}}$ denotes the Jacobian matrix of $\mathrm{Tr}(\mathbf{Q}_{n,i}\mathbf{\Psi})$ w.r.t. $\mathbf{\Psi}$ at the point $\mathbf{\Psi}^{(l)}$, and it is easy to note that the equality holds when $\mathbf{\Psi}=\mathbf{\Psi}^{(l)}$.[2]

As for the rank-one constraint, it can be equivalentlly transformed into the following form

$$\mathrm{Tr}(\mathbf{\Psi})-||\mathbf{\Psi}||_{\mathrm{s}}=0, \tag{12}$$

where $||\mathbf{\Psi}||_{\mathrm{s}}$ denotes the spectral norm of the PSD matrix $\mathbf{\Psi}$. It is noticeable that $\mathrm{Tr}(\mathbf{\Psi})=\sum_{k=1}^{K}\rho_k(\mathbf{\Psi})$ and $||\mathbf{\Psi}||_{\mathrm{s}}=\rho_1(\mathbf{\Psi})$, where $\rho_k(\mathbf{\Psi})$ indicates the k-th largest singular value of $\mathbf{\Psi}$. Hence, the equality of $\mathrm{Tr}(\mathbf{\Psi})=||\mathbf{\Psi}||_{\mathrm{s}}$ holds when the rank-one constraint is satisfied with $\rho_1(\mathbf{\Psi})>0$ and $\rho_k(\mathbf{\Psi})=0$ for $k=2,\cdots,K$, vice versa. Similarly, in the $(l+1)$-th iteration of the DC programming, a linear lower-bound of the convex item $||\mathbf{\Psi}||_{\mathrm{s}}$ at the point $\mathbf{\Psi}^{(l)}$ can be expressed as

$$\begin{aligned}||\mathbf{\Psi}||_{\mathrm{s}} &\geq ||\mathbf{\Psi}^{(l)}||_{\mathrm{s}}+\left\langle(\mathbf{\Psi}-\mathbf{\Psi}^{(l)}),\partial_{\mathbf{\Psi}}||\mathbf{\Psi}||_{\mathrm{s}}|_{\mathbf{\Psi}=\mathbf{\Psi}^{(l)}}\right\rangle \\ &\triangleq \Upsilon(\mathbf{\Psi};\mathbf{\Psi}^{(l)}),\end{aligned} \tag{13}$$

where $\partial_{\mathbf{\Psi}}||\mathbf{\Psi}||_{\mathrm{s}}|_{\mathbf{\Psi}=\mathbf{\Psi}^{(l)}}$ is a subgradient of the spectral norm $||\mathbf{\Psi}||_{\mathrm{s}}$ w.r.t. $\mathbf{\Psi}$ at the point $\mathbf{\Psi}^{(l)}$, and the equality holds when $\mathbf{\Psi}=\mathbf{\Psi}^{(l)}$. Note that one subgradient of $||\mathbf{\Psi}||_{\mathrm{s}}$ at point $\mathbf{\Psi}^{(l)}$ can be efficiently computed as $\mathbf{x}_1\mathbf{x}_1^{\mathrm{H}}$, where $\mathbf{x}_1$ is the vector corresponding to the largest singular value of $\mathbf{\Psi}^{(l)}$ [24].

With the obtained linear lower bound of $||\mathbf{\Psi}||_{\mathrm{s}}$ in (13), we can generate an approximate rank-one constraint of (12) as

$$\mathrm{Tr}(\mathbf{\Psi})-\Upsilon(\mathbf{\Psi};\mathbf{\Psi}^{(l)})\leq\varepsilon_{\Psi}, \tag{14}$$

where ε_{Ψ} is a positive threshold. The approximated rank-one constraint can guarantee that $0\leq\mathrm{Tr}(\mathbf{\Psi})-||\mathbf{\Psi}||_{\mathrm{s}}\leq\mathrm{Tr}(\mathbf{\Psi})-\Upsilon(\mathbf{\Psi};\mathbf{\Psi}^{(l)})\leq\varepsilon_{\Psi}$, and the rank-one constraint can be infinitely approaching by setting ε_{Ψ} infinitely close to zero.

Hence, we can obtain an approximation problem of ($\widetilde{\mathrm{P}}1$) at the $(l+1)$-th iteration as

$$(\mathrm{P1.1}):\max_{\mathbf{\Psi}\succeq\mathbf{0}}\ \sum_{n=1}^{N}F_{1,n}(\mathbf{\Psi})-\widehat{F}_{2,n}(\mathbf{\Psi};\mathbf{\Psi}^{(l)}) \tag{15a}$$

$$\text{s.t.}\quad \mathbf{\Psi}_{k,k}=1,\ \forall k=1,2,\cdots,K, \tag{15b}$$

$$\mathrm{Tr}(\mathbf{\Psi})-\Upsilon(\mathbf{\Psi};\mathbf{\Psi}^{(l)})\leq\varepsilon_{\Psi}, \tag{15c}$$

which is a convex optimization problem and can be readily solved by CVX [25], and the optimal solution can be obtained as $\mathbf{\Psi}^{(l+1)}$. Through choosing $\mathbf{\Psi}^{(0)}=\mathbf{\Psi}_{\chi-1}=\boldsymbol{\phi}_{\chi-1}\boldsymbol{\phi}_{\chi-1}^{\mathrm{H}}$, the feasibility of problem (P1.1) in each iteration l can always be guaranteed since $\mathbf{\Psi}^{(l-1)}$ is always a feasible solution.

Lemma 1. *The objective function of problem (P1) in* (10a) *monotonically increases with the iteration index l as*

$$\begin{aligned}&F_{1,n}(\mathbf{\Psi}^{(l+1)})-F_{2,n}(\mathbf{\Psi}^{(l+1)}) \\ \overset{(a)}{\geq}&F_{1,n}(\mathbf{\Psi}^{(l+1)})-\widehat{F}_{2,n}(\mathbf{\Psi}^{(l+1)};\mathbf{\Psi}^{(l)}) \\ \overset{(b)}{\geq}&F_{1,n}(\mathbf{\Psi}^{(l)})-\widehat{F}_{2,n}(\mathbf{\Psi}^{(l)};\mathbf{\Psi}^{(l)}) \\ =&F_{1,n}(\mathbf{\Psi}^{(l)})-F_{2,n}(\mathbf{\Psi}^{(l)}),\ \forall n\in\mathcal{N},\end{aligned} \tag{16}$$

where (a) comes from the inequality (11) *and (b) holds since $\mathbf{\Psi}^{(l)}$ is a feasible solution while $\mathbf{\Psi}^{(l+1)}$ is the optimal solution of problem (P1.1) in* (15)*. Also, the objective function of problem ($\widetilde{P}1$) is upper-bounded by UEs' limited energy budgets. Hence, Problem ($\widetilde{P}1$) in* (10) *as well as its equivalent form (P1) in* (7) *can be solved through the DC programming method with guaranteed convergence [23]. The final solution of $\mathbf{\Psi}$ at the convergence of the $(l+1)$-th iteration of the DC programming is the solution of the BCD algorithm at the χ-th iteration, i.e., $\mathbf{\Psi}_{\chi}=\mathbf{\Psi}^{(l+1)}$.*

With the obtained $\mathbf{\Psi}_{\chi}$, we can retrieve the $\boldsymbol{\phi}_{\chi}$ by decomposing $\mathbf{\Psi}_{\chi}=\boldsymbol{\phi}_{\chi}\boldsymbol{\phi}_{\chi}^{\mathrm{H}}$ and accordingly $\mathbf{\Phi}_{\chi}=\mathrm{diag}\{\boldsymbol{\phi}_{\chi}\}$. In order to facilitate the following analysis, we define the effective UE-AP channels with given $\mathbf{\Phi}$ (or $\boldsymbol{\phi}$) as

$$\mathbf{h}_n(\mathbf{\Phi})=\mathbf{H}_{\mathrm{AP}}\mathbf{\Phi}\mathbf{h}_{\mathrm{r},n},\ \forall n\in\mathcal{N}. \tag{17}$$

B. Receive Beamforming Design

With given $\mathbf{a}=\mathbf{a}_{\chi-1}$ ($p_n=a_n\widetilde{E}_n$, $n\in\mathcal{N}$) and $\mathbf{\Phi}=\mathbf{\Phi}_{\chi}$, the sub-problem for optimizing the AP's receive beamforming vextors for each UE, i,e, $\mathbf{w}_n$ for $n\in\mathcal{N}$, can be expressed as the following problem (P2)

$$(\mathrm{P2}):\max_{\mathbf{W}}\ \sum_{n=1}^{N}R_n^{\mathrm{off}}(\mathbf{w}_n), \tag{18}$$

which can be equivalently solved by addressing N parallel sub-problems for each $n\in\mathcal{N}$ as follows

$$(\mathrm{P2.1}):\max_{\mathbf{w_n}}\ \gamma_n(\mathbf{w}_n)=\frac{\mathbf{w}_n^{\mathrm{H}}\mathbf{\Theta}_n\mathbf{w}_n}{\mathbf{w}_n^{\mathrm{H}}\mathbf{\Theta}_{-n}\mathbf{w}_n}, \tag{19}$$

where $\mathbf{\Theta}_n=p_n\mathbf{h}_n(\mathbf{h}_n)^{\mathrm{H}}$ and $\mathbf{\Theta}_{-n}=\sum_{i=1,i\neq n}^{N}p_i\mathbf{h}_i(\mathbf{h}_i)^{\mathrm{H}}+\sigma^2\mathbf{I}_M$, with the effective channel $\{\mathbf{h}_n\}_{n\in\mathcal{N}}$ given in (17).

Lemma 2. *It is easy to note that problem (P2.1) in* (19) *is a generalized eigenvector problem, and its optimal solution $\mathbf{w}_n^*$ should be the eigenvector corresponds to the largest eigenvalue of the matrix $(\mathbf{\Theta}_{-n})^{-1}\mathbf{\Theta}_n$. Hence, the optimal $\mathbf{w}_n^*$ of problem (P2.1) for $n\in\mathcal{N}$ can be given as*

$$\mathbf{w}_n^*=\mathrm{eigvec}\left\{\max\left\{\mathrm{eig}\{(\mathbf{\Theta}_{-n})^{-1}\mathbf{\Theta}_n\}\right\}\right\}. \tag{20}$$

We then denote the receive beamforming matrix obtained at the χ-th iteration of the BCD algorithm as $\mathbf{W}_{\chi}=[\mathbf{w}_1^*,\cdots,\mathbf{w}_N^*]$, which is used in the following subsection.

C. Energy Partition Optimization

Here, the sub-problem (P3) for optimizing the energy partition parameters in $\mathbf{a}$ with given $\mathbf{\Phi}=\mathbf{\Phi}_{\chi}$ and $\mathbf{W}=\mathbf{W}_{\chi}$ is considered, which is given below

$$(\mathrm{P3}):\max_{\mathbf{a}}\ \sum_{n=1}^{N}\left(R_n^{\mathrm{off}}(\mathbf{a})+R_n^{\mathrm{loc}}(a_n)\right) \tag{21a}$$

$$\text{s.t.}\quad a_n\in[0,1],\ \forall n\in\mathcal{N}. \tag{21b}$$

Note that problem (P3) is non-convex because of the non-concave items $\{R_n^{\mathrm{off}}(\mathbf{a})\}_{n\in\mathcal{N}}$ in the objective function (21a). Actually, $R_n^{\mathrm{off}}(\mathbf{a})$ for $n\in\mathcal{N}$ can be re-expressed as the difference of two concave functions as follows

$$\begin{aligned}&R_n^{\mathrm{off}}(\mathbf{a})\triangleq R_{n,1}^{\mathrm{off}}(\mathbf{a})-R_{n,2}^{\mathrm{off}}(\mathbf{a}_{-n})= \\ &BT\log_2\left(\sum_{j=1}^{N}a_j\widetilde{E}_j|\mathbf{w}_n^{\mathrm{H}}\mathbf{h}_j|^2+\sigma^2||\mathbf{w}_n^{\mathrm{H}}||^2\right)- \\ &BT\log_2\left(\sum_{i=1,i\neq n}^{N}a_i\widetilde{E}_i|\mathbf{w}_n^{\mathrm{H}}\mathbf{h}_i|^2+\sigma^2||\mathbf{w}_n^{\mathrm{H}}||^2\right),\end{aligned} \tag{22}$$

[2] $\langle\mathbf{X}_1,\mathbf{X}_2\rangle\triangleq\Re\{\mathrm{tr}(\mathbf{X}_1^{\mathrm{H}}\mathbf{X}_2)\}$, where $\Re\{\cdot\}$ is the real-value operator.

where $\mathbf{a}_{-n} = [a_1, \cdots, a_{n-1}, a_{n+1}, \cdots, a_N]$.

Then the problem (P3) can also be solved with the DC programming method, where the second concave function in (22), i.e., $R_{n,2}^{\text{off}}(\mathbf{a}_{-n})$, can be substituted by its linear upper bound, so as to obtain a concave approximation of $R_n^{\text{off}}(\mathbf{a})$. Assuming $\mathbf{a}^{(m)}$ is the solution obtained at the m-th ($m = 0, 1, \cdots$) iteration of the DC programming, a linear upper bound of $R_{n,2}^{\text{off}}(\mathbf{a}_{-n})$ at the point $\mathbf{a}^{(m)}$ can be given as

$$\begin{aligned} R_{n,2}^{\text{off}}(\mathbf{a}_{-n}) &\leq \widehat{R}_{n,2}^{\text{off}}(\mathbf{a}_{-n}; \mathbf{a}_{-n}^{(m)}) \\ &= R_{n,2}^{\text{off}}(\mathbf{a}_{-n}^{(m)}) + \sum_{i=1, i\neq n}^{N} R_{n,2,i}^{\text{off}'}(\mathbf{a}_{-n}^{(m)}) * (a_i - a_i^{(m)}), \end{aligned} \tag{23}$$

where $R_{n,2,i}^{\text{off}'}(\mathbf{a}_{-n}^{(m)})$ is the first-order derivative of $R_{n,2}^{\text{off}}(\mathbf{a}_{-n})$ w.r.t a_i at the point $\mathbf{a}_{-n}^{(m)}$. Note that the equality holds when $\mathbf{a}_{-n} = \mathbf{a}_{-n}^{(m)}$. At the $(m+1)$-th iteration of DC programming, we aim at maximizing the following approximation problem

$$(\text{P3.1}): \max_{\mathbf{a}} \sum_{n=1}^{N} \left(R_{n,1}^{\text{off}}(\mathbf{a}) - \widehat{R}_{n,2}^{\text{off}}(\mathbf{a}_{-n}; \mathbf{a}_{-n}^{(m)}) + R_n^{\text{loc}}(a_n) \right) \tag{24a}$$

$$\text{s.t.} \quad a_n \in [0, 1], \ \forall n \in \mathcal{N}, \tag{24b}$$

which is a convex optimization problem and can be easily solved by CVX [25]. Through solving problem (P3.1) with CVX, the optimal solution, i.e., $\mathbf{a}^{(m+1)}$, can be finally obtained, then we have the following lemma.

Lemma 3. *The objective function of problem (P3) in* (21a) *is monotonic increasing w.r.t the iteration index* m *as*

$$\begin{aligned} &R_n^{\text{off}}(\mathbf{a}^{(m+1)}) + R_n^{\text{loc}}(a_n^{(m+1)}) \\ &\overset{(a)}{\geq} R_{n,1}^{\text{off}}(\mathbf{a}^{(m+1)}) - \widehat{R}_{n,2}^{\text{off}}(\mathbf{a}_{-n}^{(m+1)}; \mathbf{a}_{-n}^{(m)}) + R_n^{\text{loc}}(a_n^{(m+1)}) \\ &\overset{(b)}{\geq} R_{n,1}^{\text{off}}(\mathbf{a}^{(m)}) - \widehat{R}_{n,2}^{\text{off}}(\mathbf{a}_{-n}^{(m)}; \mathbf{a}_{-n}^{(m)}) + R_n^{\text{loc}}(a_n^{(m)}) \\ &= R_n^{\text{off}}(\mathbf{a}^{(m)}) + R_n^{\text{loc}}(a_n^{(m)}), \end{aligned} \tag{25}$$

where (a) *comes from the inequality* (23) *and* (b) *holds since* $\mathbf{a}^{(m)}$ *is a feasible solution while* $\mathbf{a}^{(m+1)}$ *is the optimal solution of problem (P3.1) in* (24)*. Besides, the objective* (21a) *is upper-bounded due to limited energy supply of UEs. In summary, the convergence of the proposed DC programming method for solving problem (P3) in* (21) *can be guaranteed [23]. We can obtain the final solution of* $\mathbf{a}$ *at the* χ*-th iteration of the BCD algorithm when the DC programming converges at the* $(m+1)$*-th iteration, which is denoted as* $\mathbf{a}_\chi = [a_1^{(m+1)}, \cdots, a_N^{(m+1)}]$.

IV. Location-Only Deep Learning Architecture

The proposed BCD optimization algorithm provides an optimization solution for the TCTB maximization problem (P0) in an iterative way with guaranteed convergence. However, its high computational complexity may hinder it from being applied in real-time applications. One way to overcome this drawback is leveraging the deep learning methods [16]. Through training the DNNs offline with data samples obtained from the optimization algorithms, the DNNs are capable of learning the inherent mappings of the algorithms and predict the required solutions online by mimicking the corresponding algorithms. To this end, we resort to the deep learning method in this section to obtain the solution of problem (P0).

As we mentioned before, LoS links between RIS and UEs as well as AP can be achieved by desirably deploying the RIS. In this case, the channel coefficients and the solutions of $\{\boldsymbol{\phi}, \mathbf{a}, \mathbf{W}\}$ are highly related to the locations of UEs, i.e., $\{(x_n, y_n)\}_{n\in\mathcal{N}}$. This observation facilitates the construction of a location-only deep learning architecture as shown in Fig. 1, where only UEs' locations are used as the input feature of DNNs to obtain solutions of $\{\boldsymbol{\phi}, \mathbf{a}, \mathbf{W}\}$. Note that the pilot channel estimation and feedback can be removed when utilizing this location-only architecture for online implementation.

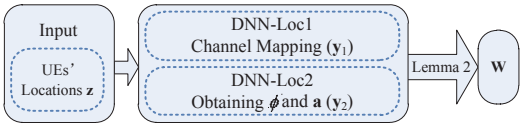

Fig. 1. Architecture for obtaining the solutions of $\{\boldsymbol{\phi}, \mathbf{a}, \mathbf{W}\}$ with the location-only DNN-Loc1 and DNN-Loc2.

Here, two DNNs are constructed. DNN-Loc1 aims at calculating the channel mapping between UEs' locations and the channel coefficients, with $2N$-dimensional input future $\mathbf{z}$ representing the UEs' locations and $I = 2(KN + MK)$-dimension output feature $\mathbf{y}_1$ representing the real and the imaginary parts of the channel coefficients $\{\mathbf{h}_{\text{r},n}\}$ and $\mathbf{H}_{\text{AP}}$. In contrast, DNN-Loc2 focuses on obtaining $\{\boldsymbol{\phi}, \mathbf{a}\}$ with input future $\mathbf{z}$ and $(K+N)$-dimensional output feature denoted as $\mathbf{y}_2 = [\widetilde{\theta}_1, \cdots, \widetilde{\theta}_K, a_1, \cdots, a_N]$, where $\{\widetilde{\theta}_k = \theta/2\pi\}_{k\in\mathcal{K}}$ are the normalized angles of the RIS reflecting coefficients $\boldsymbol{\phi}$. Then the optimal receive beamforming matrix, i.e., $\mathbf{W}$, can be easily calculated based on Lemma 2 in section III-B.

TABLE I
The Layout Parameters of DNN-Loc1 and DNN-Loc2

Parameter	DNN-Loc1	DNN-Loc2
Hidden dense layers	5	5
Neuron number	(512,512,256,128,256)	(512,256,128,64,32)
Dropout factor	(0.1,0.1,0.1,0.1,0.05)	(0.1,0.1,0.1,0.1,0.05)
Hidden activation	ELU	ELU
Output activation	Sigmoid	Sigmoid

The layouts of the feedforward DNN-Loc1 and DNN-Loc2 are given in Table I. It should be noted that we add the Batch-Normalization layer and Dropout layer between two normal dense layers to accelerate the training speed and prevent overfitting of the DNNs. Here, we adopt the function of exponential linear units (ELU) as the activation functions of the hidden layers with

$$\text{ELU}(z) = \begin{cases} z, & \text{if} \quad z > 0, \\ \alpha(\exp(z) - 1), & \text{otherwie}, \ z \leq 0. \end{cases} \tag{26}$$

which has many attractive advantages such as high learning speed, high robustness with zero-centered outputs, etc.[3] Here, the sigmoid activation function, i.e., $\text{Sigmoid}(z) = \frac{1}{1+e^{-z}}$ is leveraged at the output layers of both DNNs.

The training (including validation) and testing of the aforementioned DNNs are implemented based on the platforms of Tensorflow and Keras. Also, the adaptive moment estimation (Adam) optimizer is utilized to train the DNNs with adaptive learning rates. We adopt the mean square error (MSE) as the loss function of DNN-Loc1 for achieving the location-channel mapping, while the mean absolute error (MAE) is leveraged as the loss function of DNN-Loc2 for obtaining $\boldsymbol{\phi}$ and $\mathbf{a}$.

[3]We use the default form of ELU function in the Keras platform with α=1.

V. Simulation Results

In this section, simulation results are given to verify the effectiveness and the performance improvement of the proposed optimization algorithm and the the location-only deep learning architecture. In addition, the comparison results of the proposed optimization and deep learning methods in terms of the average running time for one realization are given to further validate the potentials of the data-driven learning approach for achieving lightweight online implementations.

A three-dimensional (3D) Euclidean coordinate system is adopted to describe the locations of the AP as $(0, y_{\rm AP}, H_0)$, the RIS as $(x_{\rm R}, 0, H_{\rm R})$ and UE $n \in \mathcal{N}$ as $(x_n, y_n, 0)$, all measured in meters (m). The aided RIS is with a uniform rectangular array (URA) of $K = K_y K_z$ reflecting elements, while the M-antenna AP is with a uniform linear array (ULA). We assume that the N ground UEs are randomly distributed in a square serving area of $D \times D$ m^2, with four vertices at horizontal locations of $(0, y_{\rm s})$, $(0, y_{\rm s}+D)$, $(D, y_{\rm s})$, and $(D, y_{\rm s}+D)$. A half-wavelength spacing is assumed among adjacent antennas/elements at the RIS and AP, then the achievable LoS channels modeled in angular domain are given as [26]

$$\mathbf{h}_{{\rm r},n} = \sqrt{L_{{\rm r},n}}\mathbf{e}^{\rm r}_{{\rm r},n}(\beta^{\rm r}_{{\rm r},n}, \gamma^{\rm r}_{{\rm r},n}),\ \forall n, \tag{27}$$

$$\mathbf{H}_{\rm AP} = \sqrt{L_{\rm AP}}\mathbf{e}^{\rm r}_{\rm AP}(\beta^{\rm r}_{\rm AP})(\mathbf{e}^{\rm t}_{\rm R}(\beta^{\rm t}_{\rm R}, \gamma^{\rm t}_{\rm R}))^{\rm H}, \tag{28}$$

where $\mathbf{e}^{\rm r}_{{\rm r},n}(\beta^{\rm r}_{{\rm r},n}, \gamma^{\rm r}_{{\rm r},n}) \in \mathbb{C}^{K\times 1} = \mathbf{e}^{\rm r}_{{\rm r},n,y}(\beta^{\rm r}_{{\rm r},n}, \gamma^{\rm r}_{{\rm r},n}) \otimes \mathbf{e}^{\rm r}_{{\rm r},n,z}(\beta^{\rm r}_{{\rm r},n}, \gamma^{\rm r}_{{\rm r},n})$ with $\mathbf{e}^{\rm r}_{{\rm r},n,y}(\beta^{\rm r}_{{\rm r},n}, \gamma^{\rm r}_{{\rm r},n}) = \{\exp(j\pi(k_y-1)\sin\beta^{\rm r}_{{\rm r},n}\sin\gamma^{\rm r}_{{\rm r},n})\}_{k_y=1}^{K_y} \in \mathbb{C}^{K_y\times 1}$ and $\mathbf{e}^{\rm r}_{{\rm r},n,z}(\beta^{\rm r}_{{\rm r},n}, \gamma^{\rm r}_{{\rm r},n}) = \{\exp(j\pi(k_z-1)\cos\beta^{\rm r}_{{\rm r},n}\sin\gamma^{\rm r}_{{\rm r},n})\}_{k_z=1}^{K_z} \in \mathbb{C}^{K_z\times 1}$, $\mathbf{e}^{\rm r}_{\rm AP}(\beta^{\rm r}_{\rm AP}) = \{\exp(j\pi(m-1)\sin\beta^{\rm r}_{\rm AP})\}_{m=1}^{M} \in \mathbb{C}^{M\times 1}$ are the receive array steering vectors with the effective angles of arrival (AOAs). Also, $\mathbf{e}^{\rm t}_{\rm R}(\beta^{\rm t}_{\rm R}, \gamma^{\rm t}_{\rm R}) \in \mathbb{C}^{K\times 1} = \mathbf{e}^{\rm t}_{{\rm R},y}(\beta^{\rm t}_{\rm R}, \gamma^{\rm t}_{\rm R}) \otimes \mathbf{e}^{\rm t}_{{\rm R},z}(\beta^{\rm t}_{\rm R}, \gamma^{\rm t}_{\rm R})$ is the transmit array steering vector with the effective angle of departure (AOD), where $\mathbf{e}^{\rm t}_{{\rm R},y}(\beta^{\rm t}_{\rm R}, \gamma^{\rm t}_{\rm R}) = \{\exp(j\pi(k_y-1)\sin\beta^{\rm t}_{\rm R}\sin\gamma^{\rm t}_{\rm R})\}_{k_y=1}^{K_y} \in \mathbb{C}^{K_y\times 1}$ and $\mathbf{e}^{\rm t}_{{\rm R},z}(\beta^{\rm t}_{\rm R}, \gamma^{\rm t}_{\rm R}) = \{\exp(j\pi(k_z-1)\cos\beta^{\rm t}_{\rm R}\sin\gamma^{\rm t}_{\rm R})\}_{k_z=1}^{K_z} \in \mathbb{C}^{K_z\times 1}$ Here, β and γ respectively represent the elevation and azimuth of AOA or AOD.

$L_{{\rm r},n}$, $L_{\rm AP}$ in (27), (28) model the distance-dependent path loss of the corresponding channels. Supposing that each element of the RIS has a 3 dBi gain due to the fact that only the front half-space reflects signals [27], then we have $L_{{\rm r},n} = 10^{0.3}L_0(d_{{\rm r},n}/d_0)^{-\alpha_{\rm r}}$ and $L_{\rm AP} = 10^{0.3}L_0(d_{\rm AP}/d_0)^{-\alpha_{\rm AP}}$ where $d_{{\rm r},n}$ and $d_{\rm AP}$ are the corresponding Euclidean distances between the transceivers, L_0 is the average constant path loss for all the channels at the reference distance of d_0, and $\alpha_{\rm r}$, $\alpha_{\rm AP}$ are the channel attenuation coefficients. The other basic simulation parameters are listed in Table II.

A. Performance Improvement

Numerical results for the proposed optimization solution ('Optimization Solution') and the location-only deep learning solution ('DL Location-Only') are presented in comparison with three benchmarks, including the 'Random RIS Setting' scheme with randomly set RIS coefficients, the 'ZF Receive Beamforming' scheme with ZF beamforming for detection, and the 'Equal Energy Allocation' scheme with equally allocated energy budgets for UEs.

In Fig. 2, we first show the TCTB of all the considered schemes w.r.t. the UEs' uniform energy budget, i.e., $E = E_n$ for $n \in \mathcal{N}$. From this figure, we can observe that the TCTB curves of all the schemes increase with E, which coincides with the intuition that more computation task-input bits can be completed if the UEs are endowed with more energy. It is clear to see that significant performance improvement can be achieved by the proposed Optimized Solution, verifying the great benefits of jointly optimizing the RIS coefficients, the receive beamforming and the UEs' energy allocation. More importantly, the DL Location-Only solution can achieve a performance very close to the proposed optimization solution, which clearly demonstrates that the proposed location-only deep learning architecture can effectively emulate the proposed BCD optimization algorithm, with a much reduced online implementation complexity.

TABLE II
Simulation Parameters

Parameter	Symbol	Value
Square area parameters	$y_{\rm s}$, D	20m, 40m
The location of the AP	$(0, y_{\rm AP}, H_0)$	(0,20,5) m
The location of the RIS	$(x_{\rm R}, 0, H_{\rm R})$	(40,0,20) m
The length of the time slot	T	5 seconds
Number of UEs	N	8
Number of AP's antenna	M	8
Number of RIS's reflecting elements	$K = K_y K_z$	24=8 × 3
Energy budget of UEs	E_n $(n \in \mathcal{N})$	10 J
Required CPU cycles per bit	C_n $(n \in \mathcal{N})$	200 cycles/bit
The effective switched capacitance	$\kappa_n (n \in \mathcal{N})$	10^{-28}
The total system bandwidth	B	40 MHz
The noise power	σ^2	-60dBm
The unit channel power gain	L_0 (d_0=1 m)	-10dB
The channel attenuation coefficients	$\alpha_{\rm r}$, $\alpha_{\rm AP}$	2.5, 2

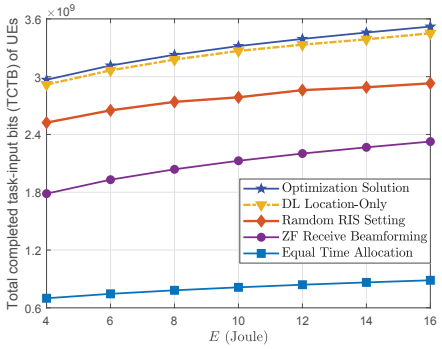

Fig. 2. The TCTB of UEs versus UEs' uniform energy budget $E = E_n$ for $n \in \mathcal{N}$.

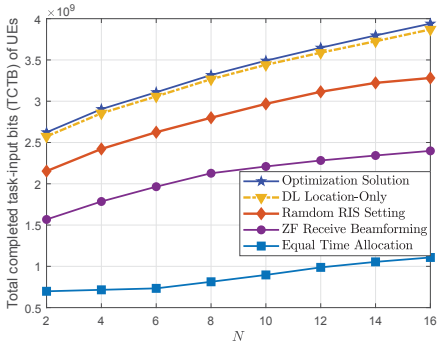

Fig. 3. The TCTB of UEs versus the number of UEs N.

We depict the results of the TCTB in Fig. 3 to study the effects of the number of UEs, i.e., N, on the system performance. Here, we further verify the effectiveness of the

DL Location-Only solution in the scenarios with different number of users, which also further demonstrates the robustness and the generalizability of the proposed location-only deep learning architecture. Similar results can be observed as from Fig. 2 that our proposed optimized solution as well as the location-only deep learning solution outperform the other benchmarks. Also, the performance gaps enlarge as N increases, indicating that better performance can be attained by the proposed methods in the cases with more UEs.

B. Implementation and Comparison

The other parameters related to training and testing the constructed location-only DNN-Loc1 and DNN-Loc2 are given in the following Table III. We present the training time and testing time of the constructed DNNs with 200000 training samples and 10000 testing samples for the case of $M = 8$, $N = 8$, and $K = 24$.[4] Here, the average running time of the BCD optimization algorithm for one realization is also given for comparison, i.e., 22.36 s, further validating the effectiveness of the location-only deep learning architecture for providing lightweight online inference solution which only require 18.36 μs for one realization.

TABLE III
PARAMETERS RELATED TO TRAINING AND TESTING

Parameter	**DNN-Loc1**	**DNN-Loc2**
Training samples	$(\mathbf{z}, \mathbf{y}_1)$	$(\mathbf{z}, \mathbf{y}_2)$
Number of training samples	200000	200000
Number of testing samples	10000	10000
Batch size	128	128
Number of epoches	1000	1000
Initial learning rate	0.001	0.001
validation split	0.2	0.2
Training time	2.6214 h	1.2238 h
Testing time	0.1836 s	0.0942 s
Average inference time	18.36 μs	9.42 μs
Average BCD Running Time	22.36 s	

VI. CONCLUSION

In this paper, a multi-user RIS-aided MEC architecture has been investigated, where the TCTB of all the UEs with limited energy supply is maximized by jointly optimizing the system parameters. A three-step BCD optimization algorithm is formulated and compared to a location-only deep learning architecture with a considerable complexity reduction. The simulation results have confirmed a close match between the performance of the BCD optimization algorithm and the location-only deep learning method, all with superior performance to the benchmarks.

ACKNOWLEDGMENT

This work is supported by the U.K. EPSRC under grant EP/S028455/1.

REFERENCES

[1] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, fourthquarter 2017.

[2] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[4]The training and testing time in Table III correspond to a computer with 64-bit Intel(R) Core(TM) i5-9600KF CPU @3.7GHz and 32 GB RAM, running Python 3.7.7 and Tensorflow 2.1.0. Note that the training and testing time can be further reduced when implemented with more powerful servers.

[3] C. You, K. Huang, H. Chae, and B. H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.

[4] X. Hu, K. Wong, and K. Yang, "Wireless powered cooperation-assisted mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 17, no. 4, pp. 2375–2388, Apr. 2018.

[5] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 856–868, Jan. 2019.

[6] X. Hu, L. Wang, K. Wong, M. Tao, Y. Zhang, and Z. Zheng, "Edge and central cloud computing: A perfect pairing for high energy efficiency and low-latency," *IEEE Trans. Wireless Commun.*, vol. 19, no. 2, pp. 1070–1083, 2020.

[7] F. Zhou, R. Q. Hu, Z. Li, and Y. Wang, "Mobile edge computing in unmanned aerial vehicle networks," *IEEE Wireless Commun.*, vol. 27, no. 1, pp. 140–146, 2020.

[8] X. Hu, K. Wong, K. Yang, and Z. Zheng, "UAV-assisted relaying and edge computing: Scheduling and trajectory optimization," *IEEE Trans. Wireless Commun.*, vol. 18, no. 10, pp. 4738–4752, Oct. 2019.

[9] X. Hu, K. Wong, and Y. Zhang, "Wireless-powered edge computing with cooperative UAV: Task, time scheduling and trajectory design," *IEEE Trans. Wireless Commun.*, pp. 1–1, 2020.

[10] Q. Wu and R. Zhang, "Towards smart and reconfigurable environment: Intelligent reflecting surface aided wireless network," *IEEE Commun. Mag.*, vol. 58, no. 1, pp. 106–112, 2020.

[11] S. Gong, X. Lu, D. T. Hoang, D. Niyato, L. Shu, D. I. Kim, and Y. C. Liang, "Towards smart wireless communications via intelligent reflecting surfaces: A contemporary survey," *IEEE Commun. Surveys Tuts.*, pp. 1–1, 2020.

[12] C. Huang, A. Zappone, G. C. Alexandropoulos, M. Debbah, and C. Yuen, "Reconfigurable intelligent surfaces for energy efficiency in wireless communication," *IEEE Trans. Wireless Commun.*, vol. 18, no. 8, pp. 4157–4170, 2019.

[13] T. Bai, C. Pan, Y. Deng, M. Elkashlan, A. Nallanathan, and L. Hanzo, "Latency minimization for intelligent reflecting surface aided mobile edge computing," *IEEE J. Sel. Areas Commun.*, pp. 1–1, 2020.

[14] S. Hua, Y. Zhou, K. Yang, and Y. Shi, "Reconfigurable intelligent surface for green edge inference," *arXiv preprint arXiv:1912.00820*, 2019.

[15] Y. Liu, J. Zhao, Z. Xiong, D. Niyato, Y. Chau, C. Pan, and B. Huang, "Intelligent reflecting surface meets mobile edge computing: Enhancing wireless communications for computation offloading," *arXiv preprint arXiv:2001.07449*, 2020.

[16] T. OShea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Trans. Cogn. Commun. Netw.*, vol. 3, no. 4, pp. 563–575, 2017.

[17] J. Chen and X. Ran, "Deep learning with edge computing: A review," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1655–1674, 2019.

[18] J. Wang, L. Zhao, J. Liu, and N. Kato, "Smart resource allocation for mobile edge computing: A deep reinforcement learning approach," *IEEE Trans. Emerg. Topics Comp.*, pp. 1–1, 2019.

[19] L. Huang, S. Bi, and Y. J. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Trans. Mobile Comp.*, pp. 1–1, 2019.

[20] H. Yang, Z. Xiong, J. Zhao, D. Niyato, L. Xiao, and Q. Wu, "Deep reinforcement learning based intelligent reflecting surface for secure wireless communications," pp. 1–1, 2020.

[21] C. Huang, R. Mo, and C. Yuen, "Reconfigurable intelligent surface assisted multiuser MISO systems exploiting deep reinforcement learning," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 8, pp. 1839–1850, 2020.

[22] X. Hu, C. Masouros, and K. Wong, "Reconfigurable intelligent surface aided mobile edge computing: From optimization-based to location-only learning-based solutions," *arXiv preprint arXiv:2102.07384*, 2021.

[23] P. D. Tao *et al.*, "The DC (difference of convex functions) programming and DCA revisited with DC models of real world nonconvex optimization problems," *Annals of operations research*, vol. 133, no. 1-4, pp. 23–46, 2005.

[24] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 2022–2035, 2020.

[25] M. Grant, S. Boyd, and Y. Ye, "CVX: Matlab software for disciplined convex programming," 2008.

[26] L. Yuan, R. Jiang, and Y. Chen, "Gain and phase autocalibration of large uniform rectangular arrays for underwater 3-d sonar imaging systems," *IEEE Journal of Oceanic Engineering*, vol. 39, no. 3, pp. 458–471, 2014.

[27] Q. Wu and R. Zhang, "Joint active and passive beamforming optimization for intelligent reflecting surface assisted swipt under qos constraints," *arXiv preprint arXiv:1910.06220*, 2019.